Université Hassan II Casablanca **M.I.P**
Faculté des Sciences et Techniques Mohammedia
Département de mathématiques

Analyse numérique

TP : Maple

Maple

Le logiciel **Maple** est un système de calcul formel. C'est un langage symbolique. C'est-à-dire que tout ce qu'il peut faire avec des nombres, il peut le faire avec des symboles.

Premiers calculs

Pour exécuter des calculs sous **Maple**, on veillera à respecter une certaine syntaxe : ils seront toujours précédés du symbole $\succ$ et seront validés par la touche **Entrée**.

$\succ$ $\mathbf{18+20; 30-14; 82*5; 12/4; 2^{100}}$;

Si on ne souhaite pas afficher des calculs intermédiaires, on pourra remplacer le symbole ; par le symbole :

$\succ$ $\mathbf{8/5^2}$:

Le symbole % sera très utile puisqu'il rappelle le dernier résultat calculé, qu'il a été affiché ou non.

$\succ$ $\mathbf{4^5}$;

$\succ$ %/**2**;

De même, le symbole %% et %%% remplacent l'avant dernier résultat exécuté et l'avant avant dernier résultat exécuté.

Des calculs formels

Maple permet d'exécuter des calculs symboliques au moyen des commandes spécifiques sous forme d'une fonction :

$\succ$ **sqrt(512)/18**; # pour calculer une racine carrée

$\succ$ **solve**($\mathbf{x^3-7\,x+6=0, x}$); # pour résoudre une équation

$\succ$ **sum**($\mathbf{1/k, k=1..n}$); # pour calculer une somme

Les nombres

- **Les nombres entiers**

  Soient a, b, n et k des entiers

  $\succ$ **ifactor**; # donne la décomposition en facteurs premiers de a

  $\succ$ **isprime(a)**; # teste la primalité de a

  $\succ$ **igcd(a, b)**; # détermine le PGCD de a et b

  $\succ$ **ilcm(a, b)**; # détermine le PPCM de a et b

  $\succ$ **iquo(a, b)**; # donne le quotient dans la division euclidienne de a et b

  $\succ$ **irem(a, b)**; # donne le reste dans la division euclidienne de a et b

  $\succ$ **n!**; # calcule la factorielle n

  $\succ$ **binomial(n, k)**; # calcule le coefficient binomial $\binom{n}{k}$

- **Les nombres rationnels**

  $\succ$ **18 + 13/6**;

- **Les nombres réels**

  $\succ$ **3 + 4 * sqrt(5)**;

  Si on souhaite obtenir une valeur approchée, on utilisera la commande **evalf**

  $\succ$ **sqrt(2)**;

  $\succ$ **evalf(sqrt(2))**; # 10 chiffres significatifs par défaut

  $\succ$ **evalf(sqrt(2), 20)**; # 20 chiffres significatifs

Les fonctions mathématiques : Fonction ou expression

Les fonctions trigonométriques :

sin(x) , *cos(x)* , *tan(x)* , *arcsin(x)* , *arccos(x)* , *arctan(x)*

Les fonctions trigonométriques hyperboliques :

sinh(x) , *cosh(x)* , *tanh(x)* , *arcsinh(x)* , *arccosh(x)* , *arctanh(x)*

Des fonctions prédéfinies:

- ≻ **ln**(**x**); # logarithme népérien de x
- ≻ **exp**(**x**); # exponentielle de x
- ≻ **sqrt**(**x**); # racine carrée de x
- ≻ **floor**(**x**); # partie entière de x
- ≻ **ceil**(**x**); # entier immédiatement supérieur à x
- ≻ **abs**(**x**); # valeur absolue de x

On peut définir de nouvelles fonctions :

- **sous forme d'une expression :**

  ≻ $\mathbf{a := (x^2 + 1)/(x - 2)};$

- **sous forme d'une fonction :**

  ≻ $\mathbf{f := x \to 3 * tan(x) - 1};$

Etude d'une fonction mathématique

- **Limites**

  ≻ $\mathbf{f := x \to sin(x)/x; limit(f(x), x = 0)};$

  ≻ $\mathbf{f := x \to x + exp(-x); limit(f(x), x = infinity)};$

  ≻ $\mathbf{limit(tan(x), x = Pi/2)};$

  ≻ $\mathbf{limit(tan(x), x = Pi/2, left); limit(tan(x), x = Pi/2, right)};$

- **Dérivées**

  ≻ $\mathbf{g := x \to x^5 - 3 * x + 7; diff(g(x), x)};$

  ≻ $\mathbf{h := x \to 2 * sin(x); D(h)};$

  ≻ **diff(h(x), x$3); (D@@3)(h);** # dérivée troisième

- **Evaluation**

  ≻ $\mathbf{f := x \to 2 * arctan(x); f(1)};$

  ≻ $\mathbf{u := (sin(x))^2(x); eval(u, x = sqrt(2)/2)};$

- **Représentation graphique**

≻ $\mathbf{v := 2 * x^2 + 3 * x - 4; plot(v, x = -5..5)};$

≻ $\mathbf{plot(v, x = -5..5, y = -10..10, discont = true)};$

Calcul intégral

≻ $\mathbf{int(sin(x) - x, x)}$; # calcul de primitive

≻ $\mathbf{int(sin(x) - x, x = 0..Pi)}$; # calcul d'intégrale définie

≻ $\mathbf{evalf(int(1/(2 * sqrt(x)), x = 0..Pi))}$; # intégrer numériquement

Equations différentielles

≻ $\mathbf{deq := diff(y(x), x\$2) - 5 * diff(y(x), x) - 6 * y(x) = 2 * exp(x)}$;

≻ $\mathbf{dsolve(deq, y(x))}$; # résolution d'une équation différentielle

Développement en séries

≻ $\mathbf{taylor(exp(-x) * sin(2 * x), x = 0, 6)}$; # développement en séries autour de 0 jusqu'au terme d'ordre 6

≻ $\mathbf{mtaylor(sin(x^2 + y^2), \{x, y\}, 8, [2, 1])}$; # développement en séries par rapport au couple de variables {x,y} jusqu'à l'ordre 8, et autour du point [2,1]

Les structures de données élémentaires

Maple nous donne les moyens de construire des structures de données :

- **Les listes**

  Pour construire une liste, il suffit d'entourer une séquence d'une paire de crochets. Les listes sont des structures ordonnées. **Maple** conserve exactement l'ordre dans lequel les éléments ont été placés dans la liste.

  ≻ $\mathbf{liste1 := [0, 2, 3, -8, 5]}$;

  ≻ $\mathbf{liste2 := [seq(i, i = 1..5)]}$; # une liste construite par la commande **seq**

  ≻ $\mathbf{liste1 + liste2}$; # ajouter la liste2 à la liste1

  ≻ $\mathbf{liste3 := [seq(i^2, i = 1..10)]}$;

  ≻ $\mathbf{liste3[8]}$; # extraire le 8 ème élément de la liste3

- **Les ensembles**

  Pour construire un ensemble, il suffit d'entourer une séquence d'une paire d'accolades {}. **Maple** élimine automatiquement les répétitions entre les {}.

  ≻ $\mathbf{ensemble1 := \{1, 2, 3, 4\}}$;

  ≻ $\mathbf{ensemble2 := \{a, b, c, d, 1, 3, 5, 3\}}$;

  Maple effectue les opérations ensemblistes habituelles :

  ≻ $ensemble1$ **union** $ensemble2$; # la réunion de deux ensembles

$\succ$ *ensemble*1 **intersect** *ensemble*2;   # l'intersection de deux ensembles

$\succ$ *ensemble*1 **minus** *ensemble*2;   # la différence de deux ensembles

L'algèbre linéaire

Pour manipuler les matrices (Matrix) et les vecteurs (Vector), il existe le package **LinearAlgebra** dans lequel on trouve un grand nombre de fonctions pour les opérations sur les matrices et les vecteurs.

$\succ$ $\mathbf{with}(\mathbf{LinearAlgebra});$

$\succ$ $\mathbf{v1} := \mathbf{Vector}([\mathbf{1}, \mathbf{5}, \mathbf{8}]);$

$\succ$ $\mathbf{v2} := \mathbf{Vector}([\mathbf{2}, -\mathbf{5}, \mathbf{3}]);$

$\succ$ $\mathbf{mat1} := \mathbf{Matrix}([[\mathbf{1}, \mathbf{2}, \mathbf{4}], [\mathbf{3}, \mathbf{5}, \mathbf{1}], [\mathbf{1}, \mathbf{0}, -\mathbf{2}]]);$

$\succ$ $\mathbf{mat2} := \mathbf{Matrix}([[\mathbf{7}, \mathbf{2}, -\mathbf{4}], [\mathbf{4}, -\mathbf{3}, -\mathbf{1}], [\mathbf{1}, \mathbf{2}, -\mathbf{2}]]);$

Pour évaluer des expressions algèbriques :

+ et **-** représentent l'addition et la soustraction

***** représente le produit par un scalaire

. représente le produit matriciel

ˆ représente l'élévation d'une matrice à une puissance entière, qui peut être négative pour l'inverse

$\succ$ $\mathbf{v1} + \mathbf{v2}; \mathbf{v1} - \mathbf{v2}; \mathbf{3} * \mathbf{v1} + \mathbf{2} * \mathbf{v2};$

$\succ$ $\mathbf{2} * \mathbf{mat1} - \mathbf{5} * \mathbf{mat2}; \mathbf{mat1}.\mathbf{mat2}; \mathbf{mat1^2}; \mathbf{mat2^{-1}};$

$\succ$ $\mathbf{Determinant}(\mathbf{mat1} - \mathbf{x} * \mathbf{IdentityMatrix}(\mathbf{3}));$ # calcule le déterminant

Sans faire appel à ce package, on peut utiliser la fonction **evalm** qui permet d'évaluer les expressions algèbriques matricielles :

$\succ$ $\mathbf{U} := \mathbf{array}([[\mathbf{1}, \mathbf{2}], [\mathbf{3}, \mathbf{4}]]);$

$\succ$ $\mathbf{V} := \mathbf{array}([[\mathbf{1}, \mathbf{1}], [\mathbf{2}, -\mathbf{1}]]);$

$\succ$ $\mathbf{evalm}(\mathbf{U} + \mathbf{2} * \mathbf{V});$

$\succ$ $\mathbf{evalm}(\mathbf{U^2});$

$\succ$ $\mathbf{evalm}(\mathbf{U^{-1}});$

$\succ$ $\mathbf{evalm}(\mathbf{sin}(\mathbf{U}));$

$\succ$ $\mathbf{evalm}(\mathbf{U}\,\&*\,\mathbf{V});$

Programmation en Maple

1. **Structures alternatives**

   if *condition* **then**
   action
   else
   action
   fi;

   Exemple

   $\succ$ $\mathbf{x := 2; y := -3}$;
   if $x > y$ **then**
   $print("Le\,minimum\,est\,y = ", y)$
   else
   $print("Le\,minimum\,est\,x = ", x)$
   fi;

2. **Structures répétitives**

   - Si le nombre de répétitions est connu d'avance, une telle structure se construit par :

     $\succ$ **for** i **from** 1 **to** 15 **do**;
     $print(i^2)$
     od;

   - Si le nombre de répétitions n'est pas connu d'avance, on recourt à :

     $\succ$ $\mathbf{x := 3}$;
     while $x < 10$ **do**
     $x := x + 2$;
     $print(x)$;
     od;

3. **Procédure**

   Une procédure est un ensemble d'instructions ordonnées. Elle se construit selon la syntaxe :

   $\succ$ **proc(donnees)**
   local (*variables locales*);
   action
   end;

Les variables locales sont utilisées durant l'exécution de la procédure mais ne sont pas accessibles à l'éxtérieur de celle-ci.

Les variables globales sont utilisées durant l'exécution de la procédure mais sont accessibles à l'éxtérieur de celle-ci.

Exemple

≻ **restart** : # réinitialisation de la session **Maple**

≻ **fact** := **proc**(**n**)
local i, p :
$p := 1$;
for i **from** 1 **to** n **do**
$p := p * i$;
od;
print("*Factorielle n* = ", p)
end;

≻ **fact**(**10**);

Si on tape p, on reçoit la lettre p

Reprenons l'exemple précédent mais déclarons la variable p comme variable globale : il suffit d'ajouter l'instruction

≻ **global p**;

Alors si on tape p, on reçoit 3628800